**Spring Data JPA - Quick Example**   
  
Create a Eclipse Project using Spring Initializr

* Go to <https://start.spring.io/>
* Change Group as “com.cognizant”
* Change Artifact Id as “orm-learn”
* In Options > Description enter "Demo project for Spring Data JPA and Hibernate"
* Click on menu and select "Spring Boot DevTools", "Spring Data JPA" and "MySQL Driver"
* Click Generate and download the project as zip
* Extract the zip in root folder to Eclipse Workspace
* Import the project in Eclipse "File > Import > Maven > Existing Maven Projects > Click Browse and select extracted folder > Finish"
* Create a new schema "ormlearn" in MySQL database. Execute the following commands to open MySQL client and create schema.

**SOLUTION:**

Creating **ormlearn** Schema in MySQL

CREATE DATABASE ormlearn;

USE ormlearn;

CREATE TABLE country (

code VARCHAR(2) PRIMARY KEY,

name VARCHAR(50));

INSERT INTO country VALUES ('IN', 'India');

INSERT INTO country VALUES ('US', 'United States of America');

USE ormlearn;

SELECT \* FROM country;

application.properties

spring.application.name=orm-learn

spring.application.name=orm-learn

logging.level.org.springframework=info

logging.level.com.cognizant=debug

logging.level.org.hibernate.SQL=trace

logging.level.org.hibernate.type.descriptor.sql=trace

logging.pattern.console=%d{dd-MM-yy} %d{HH:mm:ss.SSS} %-20.20thread %5p %-25.25logger**{25}** %25M %4L %m%n

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn

spring.datasource.username=root

spring.datasource.password=Nishanthi@3561

spring.jpa.hibernate.ddl-auto=validate

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

Package- com.cognizant.ormlearn.model

Country.java

**package** com.cognizant.ormlearn.model;

**import** jakarta.persistence.\*;

@Entity

@Table(name = "country")

**public** **class** Country {

@Id

@Column(name = "code")

**private** String code;

@Column(name = "name")

**private** String name;

**public** String getCode() {

**return** code;

}

**public** **void** setCode(String code) {

**this**.code = code;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

@Override

**public** String toString() {

**return** "Country{" + "code='" + code + '\'' + ", name='" + name + '\'' + '}';

}

}

Package- com.cognizant.ormlearn.repository

CountryRepository

**package** com.cognizant.ormlearn.repository;

**import** com.cognizant.ormlearn.model.Country;

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** org.springframework.stereotype.Repository;

@Repository

**public** **interface** CountryRepository **extends** JpaRepository<Country, String> {

}

Package - com.cognizant.ormlearn.service

CountryService.java

**package** com.cognizant.ormlearn.service;

**import** com.cognizant.ormlearn.model.Country;

**import** com.cognizant.ormlearn.repository.CountryRepository;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** org.springframework.transaction.annotation.Transactional;

**import** java.util.List;

@Service

**public** **class** CountryService {

@Autowired

**private** CountryRepository countryRepository;

@Transactional

**public** List<Country> getAllCountries() {

**return** countryRepository.findAll();

}

}

OrmLearnApplication.java (Main class)

**package** com.cognizant.ormlearn;

**import** com.cognizant.ormlearn.model.Country;

**import** com.cognizant.ormlearn.service.CountryService;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** org.springframework.context.ApplicationContext;

**import** java.util.List;

@SpringBootApplication

**public** **class** OrmLearnApplication {

**private** **static** **final** Logger ***LOGGER*** = LoggerFactory.*getLogger*(OrmLearnApplication.**class**);

**private** **static** CountryService *countryService*;

**public** **static** **void** main(String[] args) {

ApplicationContext context = SpringApplication.*run*(OrmLearnApplication.**class**, args);

*countryService* = context.getBean(CountryService.**class**);

*testGetAllCountries*();

}

**private** **static** **void** testGetAllCountries() {

***LOGGER***.info("Start");

List<Country> countries = *countryService*.getAllCountries();

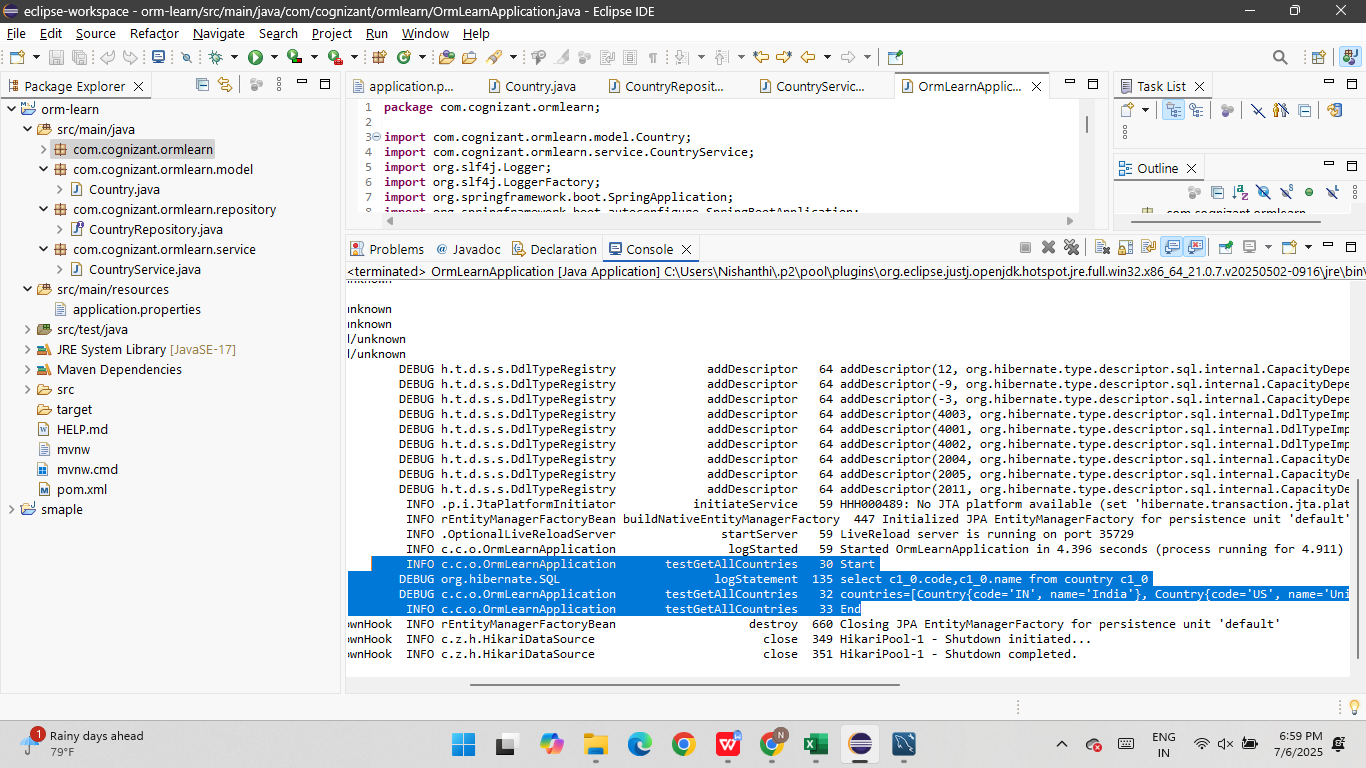
***LOGGER***.debug("countries={}", countries);

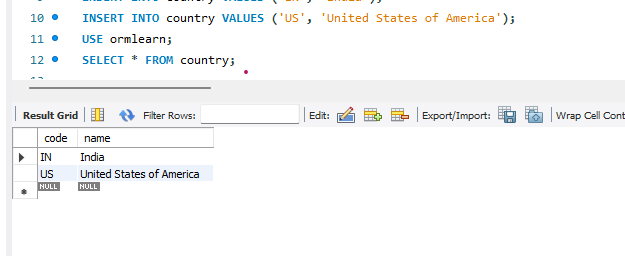
***LOGGER***.info("End");

}

}

OUTPUT:





**HANDS ON 4**

**Difference between JPA, Hibernate, and Spring Data JPA**

| **Technology** | **Explanation** |
| --- | --- |
| **JPA (Java Persistence API)** | JPA is a **set of rules** for how Java objects should be saved to a database.It just defines **what should be done**, not **how**. To use it, we need a tool like Hibernate that follows those rules. |
| **Hibernate** | Hibernate is a **real tool** that follows JPA’s rules. It does the actual work of saving, updating, and reading objects from the database. However, we need to **write more code** to manage things like database sessions and transactions. |
| **Spring Data JPA** | This is part of the Spring Framework. It sits **on top of JPA and Hibernate** to make easier. We write **less code**, and Spring handles most of the heavy lifting like saving, fetching, and even transactions — all with just a few lines! |

**CODE COMPARISON:**

**HIBERNATE:**

Session session = factory.openSession();

Transaction tx = session.beginTransaction();

session.save(employee);

tx.commit();

session.close();

**SPRING DATA JPA:**

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {}

This interface allows Spring Data JPA to automatically provide basic CRUD operations for the Employee entity using its primary key (Integer).